

PHP (autore: Vittorio Albertoni)

Premessa

Il 30 aprile 1993 il CERN decide di rendere pubblica la tecnologia alla base del World Wide Web.

Nello stesso anno nasce il primo browser, software che, installato su un computer (oggi anche un tablet o uno smartphone), ci consente di entrare nel World Wide Web per vedere che cosa contiene: azione che viene subito battezzata con il nome di «navigazione»: si chiama Mosaic. Ma il browser che prende il sopravvento, subito l'anno dopo nel 1994, è Netscape Navigator, un perfezionamento di Mosaic.

Ciò che il browser trova nel World Wide Web viene visualizzato su una pagina e si tratta, nella tecnologia di base, di documenti arricchiti da collegamenti ipertestuali con altri documenti.

La Società che sviluppava Netscape Navigator era in parte proprietà della Sun Microsystems, quella che lanciò il linguaggio di programmazione Java che, proprio in quegli stessi anni, dopo un'incubazione iniziata nel 1991, si affermava in pieno. Da subito, pertanto, vi fu un connubio tra il browser e il linguaggio Java: le così dette applet. Si trattava di programmi che, inclusi nella pagina del browser, potevano essere eseguiti utilizzando la macchina virtuale Java presente sul computer. Un modo per vivacizzare la pagina Web e instaurare processi di interazione e non solo di lettura di contenuti: su una pagina Web che promuove l'acquisto di un appartamento inserisco un programma attraverso il quale puoi calcolare quanto ti costa il mutuo per acquistarlo.

L'idea era talmente bella che Brendan Eich, nel 1995, sviluppò un nuovo linguaggio di programmazione, più facile da apprendere e da utilizzare rispetto a Java, ma, soprattutto, eseguibile attraverso un interprete incorporato nel browser, senza bisogno d'altro: per un brevissimo periodo si chiamò LiveScript ma venne subito battezzato JavaScript e il suo iniziale utilizzo, tuttora largamente diffuso, è quello di linguaggio interpretato dal browser, cioè operante sul lato client in sede di navigazione su Internet.

Ancor prima però, nel 1994, come alternativa al linguaggio Java per vivacizzare le pagine web, il danese Rasmus Lerdorf propose un linguaggio, con codice integrabile nel codice HTML di produzione della pagina, denominato PHP, inizialmente inteso come acronimo di Personal Home Page, successivamente inteso come acronimo di PHP Hypertext Preprocessor.

Mentre con l'impostazione Java/Javascript l'elaborazione del codice inserito nella pagina web avviene sul computer dell'utente, grazie alla presenza della virtual machine Java (nel caso di applet Java ormai bandite per motivi di sicurezza) o alla presenza dell'interprete Javascript nel browser (nel caso di Javascript), nel caso del codice PHP questo viene elaborato sul server e sulla pagina web destinata al browser dell'utente viene semplicemente inserito il risultato dell'elaborazione.

E' questo il motivo per cui, quando si installa l'interprete PHP sul computer, viene generalmente installato anche un server virtuale locale per produrre la pagina web contenente il risultato della elaborazione.

Ciò non toglie che possiamo vedere il risultato della elaborazione di uno script PHP anche utilizzando un interprete PHP sul terminale, rinunciando al contorno di una pagina web per contenere il risultato stesso e tutto diventa più semplice.

In questo manualetto mi propongo innanzi tutto di esporre le basi del linguaggio PHP e di chiarire i contesti in cui esso può essere applicato.

Per sapere tutto sul linguaggio PHP esiste un manuale completo in lingua italiana all'indirizzo

<https://www.php.net/manual/it/>

Indice

1	Basi del linguaggio	3
1.1	Struttura lessicale	3
1.2	Tipi di dati	3
1.2.1	Tipi scalari	3
1.2.2	Tipi composti	3
1.3	Variabili e Costanti	4
1.4	Operatori	4
1.5	Istruzioni	5
1.5.1	Istruzioni semplici	5
1.5.2	Istruzioni condizionali	6
1.5.3	Istruzioni di ciclo	7
1.6	Funzioni	8
1.6.1	Funzioni precostituite	8
1.6.2	Funzioni definite dall'utente	9
1.7	Classi e oggetti	10
2	Interprete PHP a riga di comando	11
2.1	Installazione	11
2.2	Come funziona	11
2.3	Interattività con l'utente	11
2.4	Alcuni script di esempio	12
3	PHP per la programmazione web server side	13
3.1	Installazione dell'occorrente	13
3.2	Come funziona	14
3.3	Creare la pagina	15
4	Considerazioni conclusive	17

1 Basi del linguaggio

1.1 Struttura lessicale

In PHP maiuscole e minuscole non hanno importanza quando si scrivono parole chiave del linguaggio ma il linguaggio è case-sensitive per i nomi creati dal programmatore, come i nomi di variabili, di funzioni e di classi.

Eventuali righe vuote tra una riga e l'altra di codice sono ignorate e possono invece essere utili per facilitare la comprensione del codice da parte di un lettore umano.

Il punto e virgola (;) alla fine di ciascuna istruzione è obbligatorio. Può essere omesso nel caso lo script comprenda una unica istruzione o per l'ultima istruzione di uno script comprendente più istruzioni.

I commenti al codice, se posti su una sola riga sono preceduti da //. Se sono scritti su più righe, queste sono delimitate dai caratteri /* e */.

Quando si scelgono simboli o nomi per identificare variabili o funzioni occorre evitare di utilizzare parole che hanno un loro significato nel linguaggio (le così dette parole chiave): visto che tutte queste parole sono in lingua inglese se per le nostre esigenze usiamo parole italiane evitiamo il rischio di sovrapposizione.

1.2 Tipi di dati

Per impostazione predefinita PHP è un linguaggio a tipizzazione dinamica e debole, per cui è possibile assegnare tipi di dati diversi alla stessa variabile nel corso del programma.

I tipi di dato supportati sono i seguenti.

1.2.1 Tipi scalari

int

per numeri interi.

Nei sistemi a 32 bit il massimo intero gestibile è 2.147.483.647 ($2^{32}/2 - 1$).

Nei sistemi a 64 bit il massimo intero gestibile è 9.223.372.036.854.775.807 ($2^{64}/2 - 1$).

Se un'operazione produce un numero che supera questi valori massimi PHP non genera un errore ma converte automaticamente il valore in un float (numero in virgola mobile) per preservarne la grandezza, sebbene a discapito della precisione assoluta dell'intero.

float

per numeri in virgola mobile con precisione doppia (circa 14-15 cifre decimali significative oltre le quali si perde affidabilità a causa di arrotondamenti).

Il valore massimo gestibile è comunque 1.8e308, cioè un numero di 309 cifre, di cui solo le prime 15 esatte.

string

per una sequenza di caratteri racchiusa tra apici singoli (' ') o doppi (" ").

boolean

per i valori logici true e false.

1.2.2 Tipi composti

array indicizzato

successione di valori separati da virgola tra parentesi quadre ([]) indicizzati a partire da 0.

array associativo

successione di valori tra parentesi quadre ([]) associati a chiavi in forma di stringa con l'operatore =>

Ad esempio: ["Italia" => "Roma", "Germania" => "Berlino", "Spagna" => "Madrid"]

1.3 Variabili e Costanti

Variabili e costanti sono nomi associati a un dato e identificano una zona di memoria che permette di archiviare e manipolare i dati all'interno di un programma.

Il valore contenuto in una variabile è modificabile nel corso del programma e il valore contenuto in una costante non può essere modificato.

Dal momento che PHP, come abbiamo detto è, per impostazione, un linguaggio a tipizzazione dinamica quando si crea una variabile o una costante non c'è bisogno di dichiarare il tipo di dato che deve contenere ed alla stessa variabile cui avevamo assegnato un numero possiamo, in un secondo tempo, assegnare una stringa o viceversa.

Data la mancata tipizzazione, il linguaggio converte automaticamente i valori da un tipo all'altro ove necessario.

Possiamo, per esempio, scrivere tranquillamente `3 * "4"` ed ottenere il risultato 12, cosa che mi pare sia possibile altrimenti solo nel linguaggio Javascript (prodotto tra un numero e una stringa contenente un numero).

Le variabili si creano con la sintassi

```
$<nome> = <valore>
```

dove <nome> è l'identificativo della variabile e <valore> è il valore assegnato alla variabile.

Con

```
$giudizio = "ottimo"
```

creiamo una variabile chiamata giudizio assegnandole inizialmente il valore stringa ottimo.

Con

```
$raggio = 16.5
```

creiamo una variabile chiamata raggio assegnandole inizialmente il valore float 16,5.

Le costanti si creano con la sintassi

```
define("<nome_maiuscolo>", <valore>)
```

dove <nome_maiuscolo> è l'identificativo della variabile, per convenzione scritto con caratteri maiuscoli, e <valore> è il valore assegnato alla costante.

Con

```
define("PI", 3.14159)
```

creiamo la costante PI assegnandole il valore float 3,14159.

1.4 Operatori

Gli operatori sono simboli che, inseriti tra valori o variabili contenenti valori, producono un risultato.

Operatori aritmetici

+ per la somma,

- per la sottrazione,

* per il prodotto,

/ per la divisione,

% per il modulo (resto della divisione),

** per l'elevamento a potenza (a partire da PHP 7.1)

La precedenza è quella delle espressioni algebriche (prima si eseguono elevamento a potenza, moltiplicazioni e divisioni e poi addizioni e sottrazioni). Per ottenere precedenze diverse occorre usare le parentesi.

`3+2*5` ritorna 13,

$(3+2)*5$ ritorna 25.

Operatore stringa

. concatena stringhe.

Operatori di confronto

< minore di,

<= minore o uguale a,

> maggiore di,

>= maggiore o uguale a,

== uguale a,

=== uguale a e dello stesso tipo,

!= diverso da,

!== non uguale a e di diverso tipo,

<> diverso da,

<=>

ritorna 1 se il primo termine è maggiore del secondo,

ritorna -1 se il primo termine è minore del secondo,

ritorna 0 se il primo termine è uguale al secondo.

Operatori logici

&& per l'AND logico,

|| per l'OR logico,

! per il NOT logico,

xor per il XOR logico.

1.5 Istruzioni

Il programma (nel nostro caso meglio chiamarlo script) è un insieme di istruzioni.

1.5.1 Istruzioni semplici

Sono istruzioni molto brevi, che generalmente occupano una sola riga che termina con un punto e virgola (;).

Espressioni matematiche

Sono quelle che, utilizzando valori espressi letteralmente o richiamando le variabili o le costanti che li contengono, attraverso gli operatori aritmetici determinano un nuovo valore.

Per esempio, data l'esistenza di una variabile x ,
 $32 + x - 18$; è un'espressione matematica.

Istruzioni di assegnazione

Sono quelle che assegnano a una variabile o, una sola volta, a una costante un valore espresso letteralmente o attraverso un'espressione matematica.

L'operatore di assegnazione è il segno =.

Per esempio, data l'esistenza di una variabile x ,
 $x = 22 - 9**(1/2)$; assegna a x il valore 19.

1.5.2 Istruzioni condizionali

Normalmente le istruzioni contenute nello script vengono eseguite linearmente, una dopo l'altra, nell'ordine in cui si trovano.

Attraverso l'istruzione condizionale possiamo assoggettare l'esecuzione di una o più istruzioni al verificarsi di determinate condizioni.

if

La sua sintassi prevede tre possibilità

```
if <condizione> <istruzione>
```

```
if <condizione> <istruzione> else <istruzione>
```

```
if <condizione> <istruzione> else if <condizione> <istruzione> ...else <istruzione>
```

<condizione> va scritta entro parentesi tonde;

<istruzione> va scritta entro parentesi graffe.

Se le istruzioni sono più di una vanno separate, all'interno delle parentesi graffe, da punto e virgola o a capo.

Nel primo caso se la condizione è vera viene eseguita l'istruzione, altrimenti non succede nulla e il programma continua con le altre istruzioni, se ce ne sono, non dipendenti dalla condizione.

```
if (x > y) {<istruzione_1>}
<altre_istruzioni>
```

se il valore di x è superiore al valore di y viene eseguita l'istruzione_1 e poi le altre istruzioni, in caso contrario vengono eseguite solo le altre istruzioni.

Nel secondo caso, se la condizione è vera viene eseguita la prima istruzione, altrimenti viene eseguita la seconda, quella dopo else.

```
if (x > y) {<istruzione_1>} else {<istruzione_2>}
<altre_istruzioni>
```

se il valore di x è superiore al valore di y viene eseguita l'istruzione_1 e poi le altre istruzioni, in caso contrario viene eseguita l'istruzione_2 e poi le altre istruzioni.

Nel terzo caso possiamo prevedere più casi, per esempio

```
if (<condizione>) {<istruzione_1>}
else if (<condizione>) {<istruzione_2>}
else if (<condizione>) {<istruzione_3>}
else {<istruzione_4>}
```

switch

Consente, in certi casi, di semplificare il codice quando le alternative else if viste prima sono tante.

La sintassi è

```
switch (<espressione>) {
    case <ricorrenza>: <istruzioni>; break;
    case <ricorrenza>: <istruzioni>; break;
    ...
}
```

dove

<espressione> genera valori,

<ricorrenza> è uno dei possibili valori generati da <espressione>,

<istruzioni> è ciò che si deve fare in caso di corrispondenza tra <ricorrenza> e <espressione>, l'istruzione break dopo ciascun case va inserita per interrompere l'analisi al raggiungimento della ricorrenza.

Questo programmino traduce in inglese il giorno della settimana indicato in italiano in una variabile stringa chiamata \$giorno e colloca la traduzione in una variabile chiamata \$traduzione.

```

switch ($giorno) {
    case 'Lunedì': $traduzione = 'Monday'; break;
    case 'Martedì': $traduzione = 'Tuesday'; break;
    case 'Mercoledì': $traduzione = 'Wednesday'; break;
    case 'Giovedì': $traduzione = 'Thursday'; break;
    case 'Venerdì': $traduzione = 'Friday'; break;
    case 'Sabato': $traduzione = 'Saturday'; break;
    case 'Domenica': $traduzione = 'Sunday'; break;
    default: $traduzione = 'Non ho capito il giorno'; break;
}

```

Notare la parola chiave `default`, per il caso in cui non si trovi alcuna corrispondenza tra <ricorrenza> e <espressione>.

1.5.3 Istruzioni di ciclo

Servono per ripetere più volte una istruzione o un blocco di istruzioni.

for

È il classico comando per eseguire una istruzione per un numero prefissato di volte.

La sintassi è

```

for (<partenza>; <test>; <prossimo>) <istruzione>
dove

```

<partenza> imposta il valore iniziale di un contatore a valore zero,

<test> imposta una condizione,

<prossimo> conta le ripetizioni aumentando di una unità il contatore ad ogni giro,

<istruzione> è l'azione che si deve ripetere e va scritta entro parentesi graffe.

Se le istruzioni sono più di una vanno separate, all'interno delle parentesi graffe, da punto e virgola o a capo.

Si usa identificare la variabile contatore con la lettera `i`.

```

for ($i = 0; $i < 5; $i = $i + 1) {<istruzione>}

```

esegue cinque volte l'istruzione.

while

Esegue una o più istruzioni fino a quando è vera una condizione, verificando la condizione prima di ogni iterazione.

La sintassi è:

```

while (<condizione>) <istruzione>
dove

```

<condizione> è la condizione da verificare ad ogni ciclo,

<istruzione> è ciò che si deve fare fino a quando la condizione è vera e si scrive tra parentesi graffe.

Se le istruzioni sono più di una vanno separate, all'interno delle parentesi graffe, da punto e virgola o a capo.

Se nella condizione utilizziamo un contatore, tra le istruzioni non dobbiamo dimenticare quella di aumentare di una unità il contatore stesso ad ogni giro.

```

var $i = 0
while ($i < 5) {<istruzione>; $i = $i + 1;}

```

esegue cinque volte l'istruzione.

In questo otteniamo in altro modo ciò che avevamo fatto con il comando `for`.

Ma con `while` possiamo fare altre cose, potendo esprimere condizioni non necessariamente legate a un contatore.

do...while

Esegue una o più istruzioni fino a quando è vera una condizione, verificando la condizione alla fine di ogni iterazione.

La sintassi è:

```
do <istruzione> while (condizione)
dove
```

<istruzione> è ciò che si deve fare fino a quando la condizione è vera e si scrive tra parentesi graffe.

Se le istruzioni sono più di una vanno separate, all'interno delle parentesi graffe, da punto e virgola o a capo.

<condizione> è la condizione da verificare ad ogni ciclo.

Se nella condizione utilizziamo un contatore, tra le istruzioni non dobbiamo dimenticare quella di aumentare di una unità il contatore stesso ad ogni giro.

Mentre nel ciclo `while` se la condizione non è vera già al primo ciclo non viene eseguita alcuna istruzione, nel ciclo `do...while` l'istruzione viene eseguita almeno una volta, indipendentemente dalla verifica della condizione.

1.6 Funzioni

Una funzione è un gruppo di istruzioni che è possibile richiamare tramite un nome ottenendo il compimento di un'azione o un risultato..

PHP ci offre una serie di funzioni precostituite per fare moltissime cose e ci dà anche la possibilità di costruire noi stessi altre funzioni.

1.6.1 Funzioni precostituite

Le principali funzioni precostituite presenti nella dotazione di base del linguaggio sono le seguenti. Estendendo la nostra installazione con packages aggiuntivi possiamo avere a disposizione altre funzioni, ma non ce ne occupiamo in questo manualetto in quanto entreremmo in campo altamente professionale (troviamo comunque il tutto nel manuale ufficiale all'indirizzo indicato in premessa).

funzioni algebriche

Quando si richiamano queste funzioni e le successive trigonometriche e iperboliche va indicato tra parentesi tonde l'argomento su cui esse devono agire.

L'argomento può essere un numero, una variabile contenente un numero o un'espressione matematica, nel qual caso la funzione agisce sul risultato di questa espressione.

Il tipo atteso per l'argomento è spesso `float`, ma, date le caratteristiche del linguaggio PHP, se inseriamo un intero provvede il linguaggio a convertirlo.

`abs()` per il valore assoluto

`ceil()` per arrotondare le frazioni all'intero superiore

`exp()` per calcolare l'esponente del numero e

`floor()` per arrotondare le frazioni all'intero inferiore

`intdiv()` per la divisione intera tra i due interi indicati come argomento

`log()` per il logaritmo naturale

`log10()` per il logaritmo in base 10

`pi()` per il valore approssimato di π (questa funzione non richiede argomento)

`pow()` per la potenza (gli argomenti sono base e esponente, separati da virgola)

`round()` per arrotondare un `float`

`sqrt()` per la radice quadrata

funzioni trigonometriche

Le funzioni trigonometriche dirette sono `sin()`, `cos()` e `tan()`.

L'argomento va espresso in radianti.

Le funzioni trigonometriche inverse sono `asin()`, `acos()` e `atan()`.

`deg2rad()` per convertire il numero dato in gradi nell'equivalente espresso in radianti

`rad2deg()` per convertire un numero in radianti nell'equivalente numero in gradi

funzioni iperboliche

Le funzioni iperboliche dirette sono `sinh()`, `cosh()` e `tanh()`.

Le funzioni iperboliche inverse sono `asinh()`, `acosh()` e `atanh()`

funzioni per stringhe

`strlen(<stringa>)` restituisce la lunghezza della stringa

`str_replace(<cerca>, <sostituisci>, <stringa>)` sostituisce le occorrenze di una sottostringa

`substr(<stringa>, <inizio>, <lunghezza>)` estrae una porzione di stringa

`strtolower(<stringa>)` converte in minuscolo una stringa

`strtoupper(<stringa>)` converte in maiuscolo una stringa

`trim(<stringa>)` rimuove gli spazi bianchi all'inizio e alla fine di una stringa

`ltrim(<stringa>)` rimuove gli spazi bianchi all'inizio di una stringa

`rtrim(<stringa>)` rimuove spazi bianchi alla fine di una stringa

`<stringa>[<indice>]` estrae da una stringa il carattere all'indice (0 da sinistra)

funzioni per array

`count(<array>)` restituisce il numero di elementi dell'array

`sizeof(<array>)` restituisce il numero di elementi dell'array

`in_array(<valore>, <array>)` verifica se un valore è presente nell'array

`array_pop(<array>)` rimuove l'ultimo elemento dell'array

`array_push(<array>, <valore>, <valore>...)` aggiunge valori alla fine dell'array

`array_shift(<array>)` rimuove il primo elemento dell'array

`array_unshift(<array>, <valore>, <valore>...)` aggiunge valori all'inizio dell'array

`array_merge(<array>, <array>,...)`: unisce due o più array

`array[<indice>]` estrae da un array l'elemento all'indice (0 da sinistra)

1.6.2 Funzioni definite dall'utente

Per creare una nuova funzione la sintassi è la seguente

```
function <nome> (<argomento>, <argomento>,...)  
{  
    <istruzioni>  
}
```

Una funzione che raddoppi un numero si crea così

```
function raddoppia($n)  
{  
    return $n*2;  
}
```

Per raddoppiare il numero 7 potremo così scrivere l'istruzione

```
raddoppia(7);
```

1.7 Classi e oggetti

Con quanto visto finora, se volessimo avere a disposizione uno strumento per calcolare ripetutamente circonferenza e area di cerchi di diverse misure senza dover ogni volta scrivere le relative formule per farlo potremmo costruire e salvare, in modo da poterle copiare nei nostri script, due funzioni così concepite

```
function circonferenza($raggio)
{
    return $raggio * 2 * pi();
}
```

e

```
function area_cerchio($raggio)
{
    return $raggio ** 2 * pi();
}
```

e semplicemente aggiungere nei nostri script queste istruzioni per utilizzarle

```
circonferenza(<raggio>)
```

```
area_cerchio(<raggio>)
```

E' questo il modo di procedere nella programmazione così detta funzionale, supportata dal linguaggio PHP come fin qui visto.

PHP supporta, tuttavia, anche la programmazione per oggetti e, se volessimo seguire quest'altro paradigma, dovremmo procedere così.

Innanzitutto dobbiamo prevedere una classe per poter costruire un oggetto cerchio avente le caratteristiche volute: per caratterizzare adeguatamente un oggetto "cerchio" basta il solo suo attributo "raggio".

Pertanto la classe prevede un solo attributo, il raggio:

```
class Cerchio
{
    private $raggio;
    public function __construct($raggio)
    {
        $this->raggio = $raggio;
    }
    public function calcolaArea()
    {
        return pi() * $this->raggio**2;
    }
    public function calcolaCirconferenza()
    {
        return 2 * pi() * $this->raggio;
    }
}
```

La classe indica innanzi tutto i suoi attributi: nel caso specifico uno solo, il raggio, che è quanto basta per fare i calcoli che servono per trovare circonferenza e area.

Poi occorre scrivere il metodo costruttore che è quello che consente di creare l'oggetto. E lo crea passando il suo argomento (\$raggio) all'oggetto (\$this->raggio).

Infine si indicano i due metodi (altrimenti detti funzioni membro) per calcolare area e circonferenza.

Con l'istruzione

```
$mioCerchio = new Cerchio(<raggio>);
```

creiamo poi un oggetto cerchio avente un determinato raggio e con le istruzioni

```
$mioCerchio->calcolaArea()
```

```
$mioCerchio->calcolaCirconferenza()
```

calcoliamo area e circonferenza.

2 Interprete PHP a riga di comando

Un primo modo, facile facile, di utilizzare il linguaggio PHP è quello di realizzare programmi (meglio chiamarli script, in quanto si tratta di programmi che non vengono compilati ma interpretati) e di eseguirli a terminale.

2.1 Installazione

L'interprete per il terminale si chiama **php-cli** (dove cli è l'acronimo di command line interpreter).

Se lavoriamo su Linux lo troviamo sicuramente nel repository della distro e lo possiamo installare con i comandi

```
sudo apt install php-cli
```

 se siamo su Debian e sue derivate (Ubuntu, MXLinux, Mint, ecc.)

```
sudo dnf install php-cli
```

 se siamo su Red Hat e sue derivate (Fedora).

All'indirizzo

<https://www.php.net/downloads.php>

troviamo gli strumenti per installare PHP sul sistema operativo che usiamo (Linux, Windows, Mac), inserendo la scelta nelle previste finestrelle.

Nella prima finestrella, aperta per default su WEB DEVELOPMENT, se ci interessa solo lavorare a terminale, scegliamo CLI/LIBRARY DEVELOPMENT.

2.2 Come funziona

Uno script PHP è una serie di istruzioni, scritte con il linguaggio sommariamente descritto nel precedente Capitolo, inserite tra un tag di apertura (`<?php`) e un tag di chiusura (`?>`):

```
<?php
  <istruzioni>
?>
```

Lo produciamo su un qualsiasi editor di testo (non word processor) e lo salviamo con estensione `.php`

Lo eseguiamo a terminale (quello che in Windows si chiama Prompt dei comandi) posizionandoci nella directory dove abbiamo salvato il file con il comando `php` seguito dal nome del file stesso.

Se usiamo l'editor Geany (<https://www.geany.org/>), scrivendo le istruzioni dopo aver salvato il file ancora vuoto con estensione `.php`, abbiamo il vantaggio di poterle scrivere aiutati da ottima code completion e di poter eseguire lo script direttamente dall'editor.

Con il comando a terminale

```
php -a
```

apriamo una shell interattiva in cui possiamo eseguire istruzioni PHP al volo.

2.3 Interattività con l'utente

Per gli script interpretabili a terminale ci manca sapere quali istruzioni usare perché possiamo avere a terminale i risultati delle elaborazioni (output) o perché possiamo da terminale inserire dati necessari per le elaborazioni (input).

Per l'output possiamo usare le istruzioni `echo` o `print`, seguite da ciò che va visualizzato sullo schermo: stringhe, valori di variabili richiamate con il relativo nome, risultati di espressioni matematiche.

Con `echo` possiamo indicare sulla stessa riga più cose da visualizzare separandole con virgola (`,`).

Con `print` possiamo indicare sulla stessa riga più cose da visualizzare concatenandole con l'operatore punto (`.`).

Per andare a capo dobbiamo usare la stringa `"\n"`.

Per l'input usiamo l'istruzione `readline()`, che sospende l'esecuzione dello script in attesa che l'utente inserisca il dato. Prima di questa istruzione è opportuno inserire un'istruzione con una stringa che chiarisca quale dato inserire.

2.4 Alcuni script di esempio

Cominciamo con il solito banale script in cui si chiede all'utente il nome per salutarlo:

```
<?php
echo "Come ti chiami?\n";
$nome = readline();
echo "Ciao, " . $nome . "!\n ";
?>
```

Ora uno script per calcolare il fattoriale di un numero

```
<?php
function fattoriale($n)
{
    if (($n==0)||($n==1))
    {
        return 1;
    }
    else
    {
        return fattoriale($n-1) * $n;
    }
}
echo "Indica il numero di cui vuoi calcolare il fattoriale:\n";
$n = readline();
echo "Il fattoriale di " . $n . " è:\n";
echo fattoriale($n);
?>
```

Come si vede il linguaggio PHP supporta la ricorsività.

Per quanto detto nel Paragrafo 1.2.1, questo script calcola esattamente il fattoriale per i primi venti numeri.

A partire dal fattoriale di 21 il risultato viene espresso in tipo float con le sole prime 15 cifre esatte.

Sul computer su cui sto lavorando si arriva a calcolare l'approssimazione del fattoriale di 170. Per numeri superiori viene espresso il risultato come INF, il che lascia intendere che il linguaggio PHP non contiene le insidie di altri linguaggi, come il C, che in caso di overflow non dà alcun avviso e mostra semplicemente risultati sbagliati.

* * *

Questi esempi dimostrano come il linguaggio PHP sia utilizzabile, nonostante non sia nato per questo, per creare strumenti a riga di comando, soprattutto, almeno da parte di dilettanti, per fare calcoli.

Professionalmente viene utilizzato per automatizzare task di amministrazione, come script di backup, di elaborazione di file, e altre cose varie che non richiedano interfacce grafiche.

Lavorare con PHP a terminale è anche un modo snello e semplice per provare e mettere a punto script impegnativi per la programmazione web server side, di cui parliamo nel prossimo Capitolo.

3 PHP per la programmazione web server side

Questo è il campo in cui il linguaggio PHP dà il meglio di sé: esso, infatti, è nato per vivacizzare pagine web e per rendere possibile effettuare elaborazioni online.

3.1 Installazione dell'occorrente

La scrittura del programma viene sempre fatta con un editor di testo ma per testare il programma, creato per funzionare su un server web, abbiamo bisogno del server e, per evitare complicazioni, è bene che questo server di collaudo e messa a punto non sia un vero server sul web ma sia un server virtuale collocato sul nostro computer. Questa esigenza viene soddisfatta installando il server **Apache2**, disponibile per Linux, Windows e Mac.

Poi abbiamo bisogno del linguaggio di programmazione adatto per la programmazione lato server, capace di raccogliere dati da un form, generare pagine dai contenuti dinamici, ecc. Questa esigenza viene soddisfatta con la versione CGI di PHP, il package **php-cgi**.

Ovviamente dobbiamo disporre di un **browser** web per provare l'esecuzione del programma.

Infine, per fare in modo che PHP possa lavorare a pieno regime e per qualsiasi tipo di compito, occorre uno strumento per centralizzare, gestire e rendere persistenti i dati in modo strutturato. Questa esigenza viene soddisfatta avendo a disposizione del server un Database Management System tipo MySQL, la cui versione pienamente libera oggi disponibile è **MariaDB**.

I professionisti preferiscono installare tutte queste cose una per una, configurandole a dovere anche per renderle adatte alle proprie esigenze.

Si tratta però di un lavoro non da poco e ai dilettanti cui mi rivolgo suggerisco una strada più facile.

Chi ha la fortuna di lavorare su MXLinux, una distro Linux che più bella non si può, dispone di uno strumento fenomenale, che si occupa di installare in blocco ciò che serve: basta andare su MXTOLS ▷ MX INSTALLA PROGRAMMI ▷ SERVER, selezionare LOCAL WEB SERVER e cliccare su INSTALLA.

In questo modo abbiamo una installazione in tutto simile a quella che avremmo avuto installando uno per uno gli strumenti necessari, già configurata e funzionante per normali esigenze, che vanno benissimo per un dilettante e non solo.

Su altre distro legate a Debian (Ubuntu, Mint, Mauna, ecc.) potremmo installare il tutto con l'installatore `tasksel`, ma funziona sì e no.

Per tutti gli altri, altri con sistema Linux diverso da MXLinux, altri con sistema Windows, altri con sistema Mac, esiste la possibilità di utilizzare una suite software gratuita e open-source, denominata XAMPP, che permette di creare un server web locale sul proprio computer. Il suo nome è l'acronimo di X (che significa cross-platform, in quanto funziona su Linux, Windows e macOS), Apache, MariaDB, PHP e Perl (altro linguaggio di scripting che può essere utilizzato al posto di PHP).

All'indirizzo

<https://www.apachefriends.org/it/download.html>

possiamo scegliere la versione per il nostro sistema operativo e scaricarla.

Istruzioni per l'installazione e altro si trovano cliccando sul lato destro della finestra sui link alle domande frequenti per i vari sistemi operativi.

Ad installazione avvenuta, se non abbiamo scelto diversamente durante l'installazione, troviamo tutto il software che ci serve nelle seguenti posizioni del file system:

/opt/lampp per il sistema operativo Linux,

C:\XAMPP per il sistema operativo Windows,

/Application/XAMPP per il sistema operativo Mac.

Se installiamo XAMPP e desideriamo avere anche la possibilità di utilizzare PHP a riga di comando dobbiamo installare a parte anche il package `php-cli`. Installando il software in modo diretto abbiamo a disposizione anche questa possibilità.

3.2 Come funziona

Una volta che abbiamo installato tutto quanto, per vedere se funziona dobbiamo innanzi tutto attivare il server.

Se abbiamo installato il software alla maniera normale, pezzo per pezzo, o aiutati dal tool di installazione di MXLinux che abbiamo visto nel precedente Paragrafo il server si attiva all'accensione del computer ed è sempre disponibile.

Se invece abbiamo installato XAMPP, quando abbiamo bisogno del server lo dobbiamo accendere e lo possiamo fare richiamando un'applicazione grafica in questo modo:

. se siamo su Linux, posizionati nella directory `/opt/lampp`, con il comando a terminale

```
sudo ./manager-linux-x64.run
```

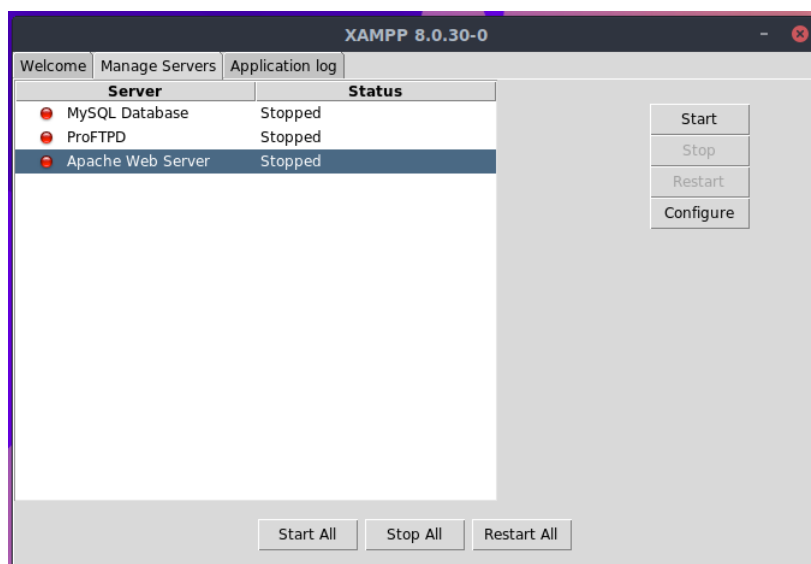
. se siamo su Windows, posizionati nella directory `C:\xampp`, con il comando a prompt dei comandi

```
xampp_control
```

. se siamo su Mac, posizionati nella directory `/Applications/XAMPP`, con il comando a terminale

```
manager-osx
```

La seguente illustrazione mostra l'applicazione grafica aperta su Linux, con selezionata la linguetta MANAGE SERVERS



Selezionata la voce APACHE WEB SERVER, che è quella che qui interessa, clicchiamo sul pulsante START e il server si accende.

Con qualche piccola differenza, abbiamo la stessa grafica sui sistemi operativi Windows e Mac.

A server acceso inseriamo nel nostro browser preferito l'indirizzo `http://localhost/` e, se tutto funziona, compare una pagina di benvenuto.

Per fare comparire una pagina creata da noi, contenente uno script PHP, dopo averla scritta nei modi che vedremo nel prossimo paragrafo, dobbiamo salvarla nella directory

. `/var/www/html` se abbiamo installato il software alla maniera normale su Linux,

oppure, se abbiamo installato XAMPP, nella sottodirectory

. `htdocs` della directory di XAMPP (lampp su Linux).

Per far comparire questa pagina dobbiamo inserirla nell'indirizzo sul browser con `http://localhost/` seguito dal nome della pagina, con estensione `.php`.

Se lavoriamo su Linux dobbiamo tener presente che le directory dove inserire le pagine sono, per default, riservate all'utente root e ciò può risultare fastidioso.

Per rimediare e potervi lavorare come utente normale basta dare il comando a terminale

```
sudo chmod -R a+rxw /var/www se abbiamo installato alla maniera normale,
```

```
sudo chmod -R a+rxw /opt/lampp se abbiamo installato XAMPP.
```

3.3 Creare la pagina

Se il nostro solo obiettivo fosse di visualizzare una scritta, con uno script del tipo

```
<?php
    echo "Ciao mondo";
?>
```

potremmo semplicemente salvare un file contenente quelle istruzioni, con estensione `.php`, ad esempio `ciao_mondo.php`, e salvarlo come abbiamo detto alla fine del paragrafo precedente. Inserendo nel browser l'indirizzo `http://localhost/ciao_mondo.php` otterremmo una pagina web nel browser contenente la scritta Ciao mondo.

Tuttavia il linguaggio PHP, destinato a vivacizzare pagine web, è stato progettato per combinarsi con l'altro linguaggio che viene utilizzato per creare pagine web, il linguaggio HTML, e possiamo allora arricchire la pagina destinata a contenere la nostra scritta con questa combinazione tra i due linguaggi.

```
<html>
  <head>
    <title>Pagina di esempio</title>
  </head>
  <body>
    <h1>Saluto prodotto con PHP</h1>
    <?php
      echo "Ciao mondo";
    ?>
  </body>
</html>
```

Salvando questo codice in un nuovo file `ciao_mondo.php` in sostituzione del precedente otteniamo sul browser



Saluto prodotto con PHP

Ciao mondo

Ovviamente occorre conoscere anche il linguaggio HTML.

Il connubio tra i due linguaggi diventa d'obbligo se lo script debba lavorare interattivamente con l'utente, per esempio per chiedergli i dati da elaborare.

Vediamo come adattare i due script presentati nel Paragrafo 2.4 ad essere interpretati sul server.

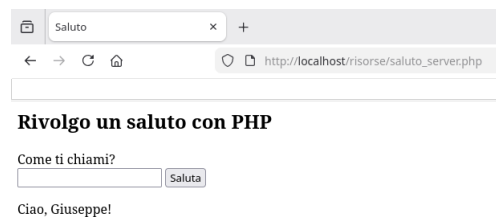
Cominciamo da quello che chiede il nome all'utente per salutarlo, che va riscritto in questo modo

```
<html>
  <head>
    <title>Saluto</title>
  </head>
  <body>
    <h2>Rivolgo un saluto con PHP</h2>
    <form method="post">
      <label>Come ti chiami?</label>
      <input type="text" name="nome">
      <button type="submit">Saluta</button>
    </form>
    <?php
      $nome = strval($_POST['nome']);
      echo "Ciao, " . $nome . "!";
    ?>
  </body>
</html>
```

Le indentazioni non sono necessarie, ma servono solo per rendere meglio leggibile il codice.

Da notare la funzione del linguaggio PHP `strval()` che legge come stringa quanto immesso nel form HTML nella variabile "nome" e lo assegna alla variabile `$nome` di PHP.

Questi sono i contenuti delle pagine web prodotte prima e dopo aver inserito il nome Giuseppe nella finestra di richiesta e premuto il tasto SALUTA.



L'altro script, per calcolare il fattoriale di un numero, va riscritto in questo modo:

```
<html>
  <head>
    <title>Calcolo Fattoriale</title>
  </head>
  <body>
    <h2>Calcola il fattoriale</h2>
    <form method="post">
      <label>Inserisci un numero intero  $\geq 0$ :</label>
      <input type="number" name="numero" min="0" required>
      <button type="submit">Calcola</button>
    </form>
    <?php
      $n = intval($_POST['numero']);
      function fattoriale($n)
      {
        if (($n==0)||($n==1))
        {
          return 1;
        }
        else
        {
          return fattoriale($n-1)*$n;
        }
      }
      echo "Il fattoriale di " . $n . " è:";
      echo "<br>";
      echo fattoriale($n);
    ?>
  </body>
</html>
```

Da notare la funzione del linguaggio PHP `intval()` che legge come numero intero quanto immesso nel form HTML nella variabile "numero" e lo assegna alla variabile `$n` di PHP.

Per leggere come float si userebbe `floatval()`.

Notare anche l'uso nel codice PHP della stringa "
" che utilizza un tag del linguaggio HTML per andare a capo.

Questi sono i contenuti delle pagine web prodotte prima e dopo aver inserito il numero 15 nella finestra di richiesta e premuto il tasto CALCOLA.



In questi esempi abbiamo la prima pagina, destinata alla raccolta dei dati, già predisposta per accogliere il risultato dell'elaborazione e questo, esteticamente, può infastidire.

Possiamo però fare in modo di avere una prima pagina esclusivamente dedicata alla raccolta dei dati ed ottenere l'elaborato su un'altra pagina esclusivamente dedicata al risultato stesso.

Per ottenere questo, riferendoci all'esempio del saluto, programmiamo la prima pagina così

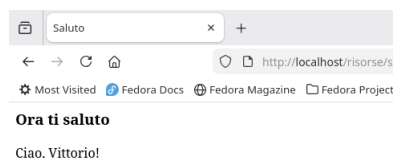
```
<html>
  <head>
    <title>Saluto</title>
  </head>
  <body>
    <h2>Rivolgo un saluto con PHP</h2>
    <form method="post" action="saluta.php">
      <label>Come ti chiami?</label>
      <input type="text" name="nome">
      <button type="submit">Saluta</button>
    </form>
  </body>
</html>
```

e, dato che non contiene codice PHP, possiamo salvarla con estensione .html.

Notare l'attributo `action="saluta.php"` inserito nel form, ad indicare lo script al quale verranno inviati i dati inseriti dall'utente per essere processati e che programmiamo così

```
<html>
  <head>
    <title>Saluto</title>
  </head>
  <body>
    <h3> Ora ti saluto </h3>
    <?php
      $nome = strval($_POST['nome']);
      echo "Ciao, " . $nome . "!";
    ?>
  </body>
</html>
```

e salviamo con estensione .php con il nome indicato nel file precedente: `saluta.php`. Indicando nel browser il nome del primo file otteniamo



A sinistra abbiamo la finestra per l'inserimento del nome della persona da salutare e a destra abbiamo la finestra di risposta che appare dopo aver inserito il nome Vittorio e aver premuto il pulsante SALUTA.

4 Considerazioni conclusive

Chi ha letto questo manualetto non è certo diventato esperto di programmazione web lato server, ma spero che almeno abbia capito di che cosa si tratta e che quanto qui imparato gli serva a poter meglio affrontare eventuali approfondimenti su testi a ciò dedicati.